APPLICATION FOR UNITED STATES PATENT

in the name of

Hugh Walsh

for

NETWORK SWITCH WITH QUALITY OF SERVICE FLOW CONTROL

9-23-03

Attorney Docket No. MP0343

EXPRESS MAIL NO.: EU984360861US

NETWORK SWITCH WITH QUALITY OF SERVICE FLOW CONTROL

Inventor: **Hugh Walsh**

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Patent Application Serial No. 60/467,873 entitled "Virtual Input Queues," filed May 5, 2003, the disclosure thereof incorporated by reference herein in its entirety.

[0002] This application is related to U.S. Non-provisional Patent Application Serial No. 10/071,417 entitled "Quality of Service Queuing System," filed February 6, 2002, U.S. Non-provisional Patent Application Serial No. 10/150,147 entitled "Apparatus and Method for Dynamically Limiting Output Queue Size in a Quality of Service Switch," filed May 17, 2002, and U.S. Non-provisional Patent Application Serial No. 10/141,096 entitled "Method and Apparatus for Preventing Blocking in a Quality of Service Switch," filed May 7, 2002, the disclosures thereof incorporated by reference herein in their entirety.

BACKGROUND

[0003] The present invention relates generally to data communications, and particularly to network switches implementing flow control for data having multiple classes of service.

[0004] The rapidly increasing popularity of networks such as the Internet has spurred the development of network services such as streaming audio and streaming video. These new services have different latency requirements than conventional network services such as electronic mail and file transfer. New quality of service (QoS) standards require that network devices, such as network switches, address these latency requirements. For example, the IEEE 802.1 standard divides network traffic into several classes of service based on sensitivity to transfer latency, and prioritizes these classes of service. The highest class of service is recommended for network control traffic, such as switch-to-switch configuration messages. The remaining classes are recommended for user traffic. The two highest user traffic classes of service are generally reserved for streaming audio and streaming video. Because the ear is more sensitive to missing data than the eye, the highest of the user traffic

classes of service is used for streaming audio. The remaining lower classes of service are used for traffic that is less sensitive to transfer latency, such as electronic mail and file transfers.

[0005] Another desirable switch feature, referred to as "flow control," prevents the switch from discarding or "dropping" frames. One flow control technique is defined by the IEEE 802.3 standard. When a switch implementing the standard becomes congested, it can exercise flow control by sending a "pause" frame to one or more link partners. In response, the link partners stop sending frames to the switch until the switch sends a "pause release" frame, or until an interval specified by the "pause" frame elapses. Thus flow control allows a switch to regulate the flow of inbound data.

[0006] However, the IEEE 802.3 flow control standard has no provision for quality of service. Therefore, when flow control is exercised on a channel under IEEE 802.3, it applies to all classes of service. High-priority traffic is delayed as long as low-priority traffic. Thus when flow control under IEEE 802.3 is imposed, quality of service cannot be achieved.

SUMMARY

[0007] In general, in one aspect, the invention features a network switching device comprising an ingress module adapted to receive frames of data from a channel, wherein each frame of data has one of a plurality of classes of service, and to store the data in one or more buffers; and an egress module adapted to exercise flow control on the channel for each of the classes of service when the number of the buffers storing frames of data received from the channel and having the class of service but not yet transmitted from the network switching device exceeds a predetermined threshold for the class of service.

[0008] Particular implementations can include one or more of the following features. To exercise flow control for one of the classes of service, the egress module is further adapted to send a pause frame to the channel, and wherein the pause frame indicates the one of the classes of service. The egress module is further adapted to terminate flow control on the channel for each of the classes of service when the number of the buffers storing frames of data having the class of service but not yet transmitted from the network switching device falls below a further predetermined threshold for the class of service. To terminate flow control for one of the classes of service, the egress module is further adapted to send a pause release frame to the channel, and wherein

the pause release frame indicates the one of the classes of service. Implementations comprise one or more queues; a forwarding module adapted to enqueue each of the buffers to one or more of the one or more queues after the ingress module stores the data of one of the frames in the buffer; a plurality of counters comprising one counter for each of the classes of service, wherein each of the counters is adapted to store a count for the channel for a respective one of the classes of service, increment the count when the forwarding module enqueues one of the buffers storing the data from one of the frames having the respective class of service, and decrement the count after the data stored in a buffer for a frame received from the channel and having the respective class of service is transmitted from the network switching device; wherein the egress module is adapted to exercise flow control on the channel for each of the classes of service when the count for the class of service exceeds the predetermined threshold for the class of service. The egress module is further adapted to terminate flow control on the channel for each of the classes of service when the count for the class of service falls below a further predetermined threshold for the class of service. Implementations comprise an integrated circuit comprising the network switching device. Implementations comprise a network switch comprising the network switching device. Implementations comprise an output-queued network switch comprising the network switching device. Implementations comprise a memory comprising the buffers. Implementations comprise a reserve module adapted to reserve one or more of the buffers to the channel; wherein the pause threshold for the channel is a function of at least one of the group consisting of the number of the buffers reserved to the channel; and the number of the buffers neither reserved nor enqueued. Implementations comprise a reserve module adapted to reserve one or more of the buffers to the channel; wherein the pause release threshold for the channel is a function of at least one of the group consisting of the number of the buffers reserved to the channel; and the number of the buffers neither reserved nor enqueued.

[0009] In general, in one aspect, the invention features a method, apparatus, and computer-readable media. It comprises receiving frames of data from a channel, wherein each frame of data has one of a plurality of classes of service; storing the data in one or more buffers; and exercising flow control on the channel for each of the classes of service when the number of the buffers storing frames of data received from the channel and having the class of service but not yet transmitted from the network switch exceeds a predetermined threshold for the class of service.

[0010] Particular implementations can include one or more of the following features. Exercising flow control for one of the classes of service comprises sending a pause frame to the channel, wherein the pause frame indicates the one of the classes of service. Implementations comprise terminating flow control on the channel for each of the classes of service when the number of the buffers storing frames of data having the class of service but not yet transmitted from the network switch falls below a further predetermined threshold for the class of service. Terminating flow control for one of the classes of service comprises sending a pause release frame to the channel, and wherein the pause release frame indicates the one of the classes of service. Implementations comprise enqueueing each of the buffers to one or more output queues after storing the data of one of the frames in the buffer; storing a count for the channel for each of the classes of service; incrementing the count for one of the classes of service when enqueueing one of the buffers storing the data from one of the frames received from the channel and having the one of the classes of service, and decrementing the count for one of the classes of service after the data stored in a buffer for a frame received from the channel and having the one of the classes of service is transmitted; wherein exercising flow control on the channel for each of the classes of service comprises exercising flow control on the channel for one of the classes of service when the count for the one of the classes of service exceeds the predetermined threshold for the one of the classes of service. Implementations comprise terminating flow control on the channel for each of the classes of service when the count for the class of service falls below a further predetermined threshold for the class of service.

[0011] In general, in one aspect, the invention features a network switching device comprising a memory adapted to store frames of data; an egress module adapted to retrieve the frames of data from the a memory, and to transmit the frames of data to a channel, wherein each of the frames of data has one of a plurality of classes of service; and an ingress module adapted to receive a pause frame indicating one or more of the classes of service to be paused; wherein, in response to the pause frame, the egress module is further adapted to cease to transmit the frames of data having the one or more classes of service to be paused, and continue to transmit the frames of data not having the one or more classes of service to be paused.

[0012] Particular implementations can include one or more of the following features. The ingress module is further adapted to receive a pause release frame indicating one or more of the classes of service to be released; wherein, in response to the pause

frame, the egress module is further adapted to resume transmitting the frames of data having the one or more classes of service to be released. Implementations comprise an integrated circuit comprising the network switching device. Implementations comprise a network switch comprising the network switching device. Implementations comprise an output-queued network switch comprising the network switch. Implementations comprise a memory comprising the buffers.

[0013] In general, in one aspect, the invention features a method, apparatus, and computer-readable media. It comprises transmitting frames of data to a channel, wherein each of the frames of data has one of a plurality of classes of service; receiving a pause frame indicating one or more of the classes of service to be paused; and in response to the pause frame, ceasing to transmit the frames of data having the one or more classes of service to be paused, and continuing to transmit the frames of data not having the one or more classes of service to be paused.

[0014] Particular implementations can include receiving a pause release frame indicating one or more of the classes of service to be released; and in response to the pause frame, resuming transmitting the frames of data having the one or more classes of service to be released.

[0015] The details of one or more implementations are set forth in the accompanying drawings and the description below. Other features will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

[0016] FIG. 1 shows a simple network in which a network switch connects two devices.

[0017] FIG. 2 is a block diagram of a conventional shared-memory output-queue store-and-forward network switch that can act as the switch in network of FIG. 1.

[0018] FIG. 3 is a flowchart of a conventional process performed by the network switch of FIG. 2.

[0019] FIG. 4 is a block diagram of a queue controller suitable for use as the queue controller in the network switch of FIG. 2.

[0020] FIG. 5 depicts the manner in which these pointers circulate within the queue controller of FIG. 4.

[0021] FIG. 6 is a block diagram of an output queue according to one implementation.

[0022] FIGS. 7A and 7B show a flowchart of a process of a network switch such as the switch of FIG. 2 under control of the queue controller of FIG. 4 according to one implementation.

[0023] The leading digit(s) of each reference numeral used in this specification indicates the number of the drawing in which the reference numeral first appears.

DETAILED DESCRIPTION

[0024] FIG. 1 shows a simple network 100 in which a network switch 102 connects two devices 104A and 104B. Each of devices 104 can be any network device, such as a computer, a printer, another network switch, or the like. Switch 102 transfers data between devices 104 over channels 106A and 106B, and can also handle an arbitrary number of devices in addition to devices 104. Channels 106 can include fiber optic links, wireline links, wireless links, and the like.

[0025] FIG. 2 is a block diagram of a conventional shared-memory output-queue store-and-forward network switch 200 that can act as switch 102 in network 100 of FIG. 1. Switch 200 has a plurality of ports including ports 202A and 202N. Each port 202 is connected to a channel 204, a queue controller 206 and a memory 208. Each port 202 includes an ingress module 214 that is connected to a channel 204 by a physical layer (PHY) 210 and a media access controller (MAC) 212. Referring to FIG. 2, port 202A includes an ingress module 214A that is connected to channel 204A by a MAC 212A and a PHY 210A, while port 202N includes an ingress module 214N that is connected to channel 204N by a MAC 212N and a PHY 210N. Each port 202 also includes an egress module 216 that is connected to a channel 204 by a MAC 218 and a PHY 220. Referring to FIG. 2, port 202A includes an egress module 216A that is connected to channel 204A by a MAC 218A and a PHY 220A, while port 202N includes an egress module 216N that is connected to channel 204N by a MAC 218N and a PHY 220N.

[0026] FIG. 3 is a flowchart of a conventional process 300 performed by network switch 200. At power-on, queue controller 206 initializes a list of pointers to unused buffers in memory 208 (step 302). A port 202 of switch 200 receives a frame from a channel 204 (step 304). The frame enters the port 202 connected to the channel 204 and traverses the PHY 210 and MAC 212 of the port 202 to reach the ingress module

214 of the port 202. Ingress module 214 requests and receives one or more pointers from queue controller 206 (step 306). Ingress module 214 stores the frame at the buffers in memory 208 that are indicated by the received pointers (step 308).

[0027] Ingress module 214 then determines to which channel (or channels in the case of a multicast operation) the frame should be sent, according to methods well-known in the relevant arts (step 310). Queue controller 206 sends the selected pointers to the egress modules 216 of the ports connected to the selected channels (step 312). These egress modules 216 then retrieve the frame from the buffers indicated by the pointers (step 314) and send the frame to their respective channels 204 (step 316). These egress modules 216 then release the pointers for use by another incoming frame (step 318). The operation of switch 200 is termed "store-and-forward" because the frame is stored completely in the memory 208 before leaving the switch 200. The store-and-forward operation creates some latency. Because all of the switch ports 202 use the same memory 208, the architecture of switch 202 is termed "shared memory."

[0028] The queue controller 206 performs the switching operation by operating only on the pointers to memory 208. The queue controller 206 does not operate on the frames. If pointers to frames are sent to an egress module 216 faster than that egress module 216 can transmit the frames over its channel 204, the pointers are queued within that port's output queue 216. Because pointers accumulate only at the output side of switch 200, the architecture of switch 200 is also termed "output-queued." Thus switch 200 has a store-and-forward, shared-memory, output-queued architecture.

[0029] FIG. 4 is a block diagram of a queue controller 400 suitable for use as queue controller 206 in network switch 200 of FIG. 2. Queue controller 400 can be implemented using hardware, software, or any combination thereof. Queue controller 400 includes a forwarding module 402, a free module 404, a plurality of reserve modules 406A through 406N, a plurality of virtual queue modules (VQM) 416A through 416N, and a plurality of output queues 408A through 408N. Each reserve module 406 and virtual queue module 416 is connected to one of ingress modules 214. Each output queue 408 is connected to one of egress modules 216.

[0030] Each virtual queue module 416 comprises a plurality of counters, one for each class of service of the frames of data received by network switch 200. Each counter keeps a count of the number of buffers storing frames of data that (1) have the class of service of the counter, (2) were received on the channel 204 served by the counter

and enqueued to one of output queues 408, and (3) have not yet been transmitted by switch 200. The counts kept by these counters are used to implement quality of service flow control, as described in detail below.

[0031] Free module 404 and reserve modules 406 each contain one linked list of pointers to buffers in shared memory 208. Each output queue 408 contains a priority queue for each class of service implemented by switch 400. Each priority queue contains one linked list of pointers to buffers in shared memory 208. In one implementation, switch 400 implements four classes of service labeled class 0 through class 3, with class 3 having the highest priority. In this implementation, each output queue 408 contains four priority queues. Other implementations can implement fewer or greater classes of service, as will be apparent to one skilled in the relevant art after reading this description.

[0032] All of the linked lists for free module 404, reserve modules 406, and output queues 408 are stored in a linked-list memory 410. A memory arbiter 412 arbitrates among competing requests to read and write linked-list memory 410. Each of free module 404, reserve modules 406, and output queues 408 maintains an object that describes its linked list. Each of these objects maintains the size of the list and pointers to the head and tail of the list. Each of free module 404, reserve modules 406, and output queues 408 traverses its linked list by reading and writing the "next" links into and out of linked list memory 410.

[0033] Free module 404 contains pointers to buffers in memory 208 that are available to store newly-received frames (that is, the buffers have an available status). Each reserve module 406 contains a list of pointers to available buffers that are reserved for the port housing that reserve module. FIG. 5 depicts the manner in which these pointers circulate within queue controller 400. Queue controller 400 allocates pointers from free module 404 to reserve modules 406 according to the methods described below (flow 502). Buffers associated with pointers in a free module 404 have an available status until a frame is stored in the buffers. Storing a frame in one or more buffers changes the status of those buffers to unavailable. To forward a frame to an output port, the frame is stored in a buffer in memory 208, and the pointers to that buffer are transferred to the output queue 408 for that output port (flow 504). When a frame is sent from an output port to a channel 106, the pointers for that frame are returned to free module 404, thereby changing the status of the pointers to available (flow 506).

[0034] Multicast module 414 handles multicast operations. In linked-list memory 410, pointers associated with the start of a frame also have a vector including a bit for each destined output port for the frame. When an output port finishes transmitting a frame, the output queue passes the frame's pointers to multicast module 414, which clears the bit in the destination vector associated with that output port. When all of the bits in the destination vector have been cleared, the frame's pointers are returned to free module 404.

[0035] FIG. 6 is a block diagram of an output queue 408 according to one implementation. Output queue 408 includes an output scheduler 602 and four priority queues 604A, 604B, 604C, and 604D assigned to classes of service 3, 2, 1, and 0, respectively. Forwarding module 402 enqueues the pointers for each frame to a priority queue selected according to the class of service of the frame. For example, the pointers for a frame having class of service 2 are enqueued to priority queue 604B. Each egress module 216 can transmit only one frame at a time. Therefore output scheduler 602 selects one of the priority queues at a time based on a priority scheme that can be predetermined or selected by a user of the switch, such as a network administrator.

[0036] One priority scheme is strict priority. According to strict priority, higher-priority frames are always handled before lower-priority frames. Under this scheme, priority queue 604A transmits until it empties. Then priority queue 604B transmits until it empties, and so on.

[0037] Another priority scheme is weighted fair queuing. According to weighted fair queuing, frames are processed so that over time, higher-priority frames are transmitted more often than lower-priority frames according to a predetermined weighting scheme and sequence. One weighting scheme for four classes of service is "8-4-2-1." Of course, other weighting schemes can be used, as will be apparent to one skilled in the relevant art after reading this description.

[0038] According to 8-4-2-1 weighting, in 15 consecutive time units, 8 time units are allocated to class of service 3, 4 time units are allocated to class of service 2, 2 time units are allocated to class of service 1, and 1 time unit is allocated to class of service 0. In one implementation, the sequence shown in Table 1 is used with 8-4-2-1 weighting.

[0039]

Time Unit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Priority	3	2	3	1	3	2	3	0	3	2	3	1	3	2	3

Table 1

[0040] Thus when none of the priority queues are empty, the sequence of classes of service selected by output scheduler 602 is 3-2-3-1-3-2-3-0-3-2-3-1-3-2-3. When one of the priority queues is empty, its slots in the sequence are skipped. For example, if only priority queue 604A is empty, the sequence of classes of service selected by output scheduler 602 is 2-1-2-0-2-1-2.

[0041] FIGS. 7A and 7B show a flowchart of a process 700 of a network switch such as switch 200 under control of queue controller 400 according to one implementation. At power-on of switch 200, queue controller 400 initializes a free module 404 to contain a number of pointers to unused buffers in memory 208, and initializes the counters in virtual queue modules 416 to zero (step 702). Queue controller 400 transfers some of these pointers to each reserve module 406 (step 704).

[0042] Each reserve module 406 includes a counter to count the number of pointers in the reserve module. When the number of pointers is below the capacity of the reserve module 406, the reserve module continually requests pointers from free module 404 (step 706). In some implementations, the capacity of each reserve module 406 is 4 pointers, where a frame of maximum size requires 3 pointers.

[0043] A port 202 of switch 200 receives a frame from a channel 204 (step 708). The frame enters the port 202 connected to the channel 204 and traverses the PHY 210 and MAC 212 of the port 202 to reach the ingress module 214 of the port 202. Ingress module 214 receives one or more pointers from the reserve module 406 for the port 202 (step 710). A frame data memory controller within ingress module 214 stores the frame in memory 208 at the buffers that are indicated by the received pointers (step 712). Ingress module 214 then determines the destination channel (or channels in the case of a multicast operation) to which the frame should be sent, according to methods well-known in the relevant arts (step 714).

[0044] Forwarding module 402 then enqueues the buffers for the frame to the destination channels of the frame (step 716). Forwarding module 402 enqueues the buffers by sending the pointers for the buffers to the output queues 408 for the ports connected to the destination channels.

[0045] The counter for the class of service of the frame in the virtual queue module 416 associated with the ingress module 214 storing the frame in the buffers increments once for each buffer enqueued for data received by that ingress module, preferably after each buffer is enqueued in order to maintain an accurate count. In some embodiments, the counter increments when the corresponding reserve module sends a pointer to forwarding module 402. In other embodiments, the counter increments only after forwarding module 402 has sent the pointer to all of its destination output queues 408. When the count of any counter exceeds a "pause" threshold Pon (step 718), the corresponding egress module 216 exercises flow control on the corresponding channel for the class of service of the counter (step 720). Of course, each class of service can have a different pause threshold Pon.

[0046] In some embodiments, the pause threshold for each counter is offset by the number of buffers reserved by the corresponding reserve module 406 such that the corresponding egress module 216 exercises flow control on a channel for a class of service when the count of the corresponding counter exceeds the pause threshold less the number of buffers reserved by the corresponding reserve module 406.

[0047] In some embodiments, a dynamic pause threshold is used, for example based on the number of pointers in free module 404. For example, the dynamic pause threshold Pondyn could be determined by

[0048] $$Pondyn = Kon \times FreeSize \pm Offset \quad (1)$$

[0049] where Kon and Offset are constants and FreeSize is the number of pointers in free module 404.

[0050] In an implementation where a port 202 is connected to a full-duplex channel, the port 204 exercises flow control on the channel by sending a "pause" frame to the channel, and releases flow control by sending a "pause release" frame to the channel. The pause and pause release frames both comprise a bit vector having a bit for each class of service. When a bit is set in the bit vector in a pause frame, it indicates traffic for the corresponding class of service should be paused. Similarly, when a bit is clear in the bit vector in a pause release frame, it indicates traffic for the corresponding class of service can resume. This use of a bit vector allows a single pause or pause release frame to exercise or terminate flow control for multiple classes of service.

[0051] When the pointers for the frame reach the head of an output queue 408 of a port 202, the egress module 216 of the port retrieves the frame from the buffers indicated by the pointers (step 728) and sends the frame to its channel 204 (step 730).

The output queue 408 then releases the pointers by returning them to free module 404 (step 732).

[0052] The counter for the class of service of the frame in the virtual queue module 416 associated with the ingress module 214 that originally received the frame just transmitted decrements once for each buffer of data transmitted for the frame (step 734), preferably as each buffer is freed in order to maintain an accurate count. When the count of any counter falls below a "pause release" threshold Poff (step 736), the corresponding egress module 216 terminates flow control on the corresponding channel for the class of service of the counter (step 738). Of course, each class of service can have a different pause release threshold Poff.

[0053] In some embodiments, the pause release threshold for each counter is offset by the number of buffers reserved by the corresponding reserve module 406 such that the corresponding egress module 216 terminates flow control on a channel for a class of service when the count of the corresponding counter falls below the pause release threshold less the number of buffers reserved by the corresponding reserve module 406.

[0054] In some embodiments, a dynamic pause threshold is used, for example based on the number of pointers in free module 404. For example, the dynamic pause threshold Poffdyn could be determined by

[0055] $$Poffdyn = Koff \times FreeSize \pm Offset \qquad (2)$$

[0056] where Koff and Offset are constants and FreeSize is the number of pointers in free module 404. Process 700 then resumes at step 706.

[0057] Any combination of static and dynamic thresholds, whether offset by the number of buffers reserved by the reserve module or not, can be used for exercising or terminating flow control on a channel.

[0058] Each counter is decremented in the following manner. When a reserve module 406 forwards a pointer to an output queue 408, it writes the class of service of the corresponding frame, a source port identifier (SPID) and a destination port vector (DPV) to a header field of the pointer. The DPV is preferably an n-bit vector having a bit for each port 202 of switch 102. Each bit set to one in the DPV indicates a corresponding port 202 as a destination for the data stored in the buffer identified by the pointer.

[0059] As described above, each output queue releases a pointer after transmitting the data in the buffer identified by the pointer. When a pointer is released by an output queue 408, multicast module 414 sets the bit for that output queue in the DPV for the released pointer to zero. When a DPV becomes all-zero, indicating that the corresponding data has been transmitted to all of its destination channels, multicast module 414 causes the counter for the class of service identified in the pointer header in the virtual queue module 416 in the port 202 identified by the SPID for the pointer to decrement.

[0060] By maintaining virtual input queues (in the form of the counters in virtual input queue modules 416), embodiments of the present invention achieve the accurate and rapid flow control of an input-queued switch in a high-performance output-queued switch while preserving quality of service.

[0061] The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor; and method steps of the invention can be performed by a programmable processor executing a program of instructions to perform functions of the invention by operating on input data and generating output. The invention can be implemented advantageously in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each computer program can be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language can be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Generally, a computer will include one or more mass storage devices for storing data files; such devices include magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and optical disks. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-

optical disks; and CD-ROM disks. Any of the foregoing can be supplemented by, or incorporated in, ASICs (application-specific integrated circuits) .

[0062] A number of implementations of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. Please list any additional modifications or variations. Accordingly, other implementations are within the scope of the following claims.